(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
1 March 2001 (01.03.2001)

PCT

(10) International Publication Number
WO 01/14987 A2

(51) International Patent Classification⁷: G06F 15/16

(21) International Application Number: PCT/US00/22783

(22) International Filing Date: 17 August 2000 (17.08.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/150,394 23 August 1999 (23.08.1999) US
09/502,170 11 February 2000 (11.02.2000) US

(71) Applicant: TERRASPRING, INC. [US/US]; 48800 Milmont Drive, Fremont, CA 94538 (US).

(72) Inventors: AZIZ, Ashar; 4180 Tanager Common, Fremont, CA 94555 (US). MARKSON, Tom; 30 Mounds Road #206, San Mateo, CA 94402 (US). PATTERSON, Martin; 1445 Mercy Street, Mountain View, CA 94041 (US).

(74) Agents: HICKMAN, Brian et al.; Hickman Palermo Truong & Becker, 1600 Willow Street, San Jose, CA 95126 (US).

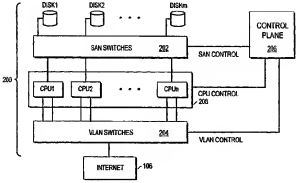
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *Without international search report and to be republished upon receipt of that report.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: EXTENSIBLE COMPUTING SYSTEM



(57) Abstract: Methods and apparatus providing a dynamically sized, highly scalable and available server farm are disclosed. A Virtual Server Farm (VSF) is created out of a wide scale computing fabric ("Computing Grid") which is physically constructed once and then logically divided up into VSFs for various organizations on demand. Each organization retains indcpendent administrative control of a VSF. A VSF is dynamically firewalled within the Computing Grid. An allocation and control of the elements in the VSF is performed by a Control Plane connected to all computing, networking, and storage elements in the computing grid through special control ports. The internal topology of each VSF is under control of the Control Plane. No physical rewiring is necessary in order to construct VSFs in many different configurations, including single-tier Web server or multi-tier Web-server, application server, database server configurations. Each tier of a multi-tier VSF (e.g. Web server tier, application server tier, database server tier, etc) can be dynamically sized based on the load on the servers in that particular tier. Storage devices may include a plurality of pre-defined logical blueprints that are associated with roles that may be assumed by the computing grid elements. Initially, no computing element is dedicated to any particular role or task such as Web server, application server, database server, etc. The role of each computing element is acquired from one of a plurality of pre-defined, stored blueprints, each of which defines a boot image for the computing elements that are associated with that role.

WO 01/14987 A2

EXTENSIBLE COMPUTING SYSTEM

FIELD OF THE INVENTION

The present invention generally relates to data processing. The invention relates more specifically to methods, apparatus, and mechanisms providing an extensible, flexible, and scalable computing system.

BACKGROUND OF THE INVENTION

Builders of Web sites and other computer systems today have to deal with many systems planning issues. These include capacity planning for normal growth, expected or unexpected peak demand, availability and security of the site, etc. Companies who wish to provide services on the Web have new business and service models, which are the areas in which they want to innovate and lead, but in order to do so they have to deal with the non-trivial complexity of designing, building and operating a large-scale Web site. This includes the need to grow and scale the site while it is operational.

Doing all this requires finding and hiring trained personnel capable of engineering and operating such a site, which may be potentially large and complicated. This is creating difficulty for many organizations, because designing, constructing and operating such large sites is simply not their core competency.

One response to these issues is to host an enterprise Web site at a third party site, co-located with other Web sites of other enterprises. Such outsourcing facilities are currently available from companies such as Exodus, AboveNet, GlobalCenter, etc. These facilities provide physical space, and redundant network and power facilities so that the enterprise customer or user need not provide them. The network and power facilities are shared among many enterprises or customers.

However, the users of these facilities are still required to do a lot of work relating to their computing infrastructure in the course of building, operating and growing their facilities. Information technology managers of the enterprises hosted at such facilities remain responsible for selecting, installing, configuring, and maintaining their own

computing equipment at the facilities. The managers must still confront difficult issues such as resource planning and handling peak capacity.

Even when outsourcing companies also provide computing facilities (e.g., Digex), the facilities are no easier to scale and grow for the outsourcing company, because growth involves the same manual and error-prone administrative steps. In addition, problems remain with capacity planning for unexpected peak demand.

Further, each Web site may have different requirements. For example, particular Web sites may require the ability to be independently administered and controlled. Others may require a particular type or level of security that isolates the Web site from all other sites that are co-located at the service provider. Others may require a secure connection to an enterprise Intranet located elsewhere.

Also, various Web sites differ in internal topology. Some sites simply comprise a row of Web servers that are load balanced by a Web load balancer. Suitable load balancers are Local Director from Cisco Systems, Inc., BigIP from F5Labs, Web Director from Alteon, etc. Other sites may be constructed in a multi-tier fashion, whereby a row of Web servers handle Hypertext Transfer Protocol (HTTP) requests, but the bulk of the application logic is implemented in separate application servers. These application servers in turn may need to be connected back to a tier of database servers.

Some of these different configuration scenarios are shown in FIG. 1A, FIG. 1B, and FIG. 1C. FIG. 1A is a block diagram of a simple Web site, comprising a single machine 100 comprising a CPU 102 and disk 104. Machine 100 is coupled to the global, packet-switched data network known as the Internet 106, or to another network. Machine 100 may be housed in a co-location service of the type described above.

FIG. 1B is a block diagram of a 1-tier Web server farm 110 comprising a plurality of Web servers WSA, WSB, WSC. Each of the Web servers is coupled to a load-balancer 112 that is coupled to Internet 106. The load balancer divides the traffic between the servers to maintain a balanced processing load on each server. Load balancer 112 may also include or may be coupled to a firewall for protecting the Web servers from unauthorized traffic.

FIG. 1C shows a 3-tier server farm 120 comprising a tier of Web servers W1, W2, etc., a tier of application servers A1, A2, etc., and a tier of database servers D1, D2, etc. The Web servers are provided for handling HTTP requests. The application servers execute the bulk of the application logic. The database servers execute database management system (DBMS) software.

Given the diversity in topology of the kinds of Web sites that may need to be constructed, it may appear that the only way for constructing large-scale Web sites is to custom build each one. Indeed, this is the conventional approach. Many organizations are separately struggling with the same issues, and custom building each Web site from scratch. This is inefficient and involves a significant amount of duplicate work at different enterprises.

Still another problem with the conventional approach is resource and capacity planning. A Web site may receive vastly different levels of traffic on different days or at different hours within each day. At peak traffic times, the Web site hardware or software may be unable to respond to requests in a reasonable time because it is overloaded. At other times, the Web site hardware or software may have excess capacity and be underutilized. In the conventional approach, finding a balance between having sufficient hardware and software to handle peak traffic, without incurring excessive costs or having over-capacity, is a difficult problem. Many Web sites never find the right balance and chronically suffer from under-capacity or excess capacity.

Yet another problem is failure induced by human error. A great potential hazard present in the current approach of using manually constructed server farms is that human error in configuring a new server into a live server farm can cause the server farm to malfunction, possibly resulting in loss of service to users of that Web site.

Based on the foregoing, there is a clear need in this field for improved methods and apparatus for providing a computing system that is instantly and easily extensible on demand without requiring custom construction.

There is also a need for a computing system that supports creation of multiple segregated processing nodes, each of which can be expanded or collapsed as needed to

account for changes in traffic throughput. Other needs will become apparent in the disclosure provided in this document.

SUMMARY OF THE INVENTION

The foregoing needs and objects, and other needs and objects that will become apparent from the following description, are achieved by the present invention, which comprises, in one aspect, a method and apparatus for creating highly scalable, highly available and secure data processing sites, based on a wide scale computing fabric ("computing grid"). The computing grid is physically constructed once, and then logically divided up for various organizations on demand. The computing grid comprises a large plurality of computing elements that are coupled to one or more VLAN switches and to one or more storage area network (SAN) switches. A plurality of storage devices are coupled to the SAN switches and may be selectively coupled to one or more of the computing elements through appropriate switching logic and commands. One port of the VLAN switch is coupled to an external network, such as the Internet. A supervisory mechanism, layer, machine or process is coupled to the VLAN switches and SAN switches.

Initially, all storage devices and computing elements are assigned to Idle Pools. Under program control, the supervisory mechanism dynamically configures the VLAN switches and SAN switches to couple their ports to one or more computing elements and storage devices. As a result, such elements and devices are logically removed from the Idle Pools and become part of one or more virtual server farms (VSFs). Each VSF computing element is pointed to or otherwise associated with a storage device that contains a boot image usable by the computing element for bootstrap operation and production execution.

By physically constructing the computing grid once, and securely and dynamically allocating portions of the computing grid to various organizations on demand, economies of scale are achieved that are difficult to achieve when doing a custom build of each site.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings and in which like reference numerals refer to similar elements and in which:

FIG. 1A is a block diagram of a simple Web site having a single machine topology.

FIG. 1B is a block diagram of a one-tier Web server farm.

FIG. 1C is a block diagram of a three-tier Web server farm.

FIG. 2 is a block diagram of one configuration of an extensible computing system 200 that includes a local computing grid.

FIG. 3 is a block diagram of an exemplary virtual server farm featuring a SAN Zone.

FIG. 4A, FIG. 4B, FIG. 4C, and FIG. 4D are block diagrams showing successive steps involved in adding a computing element and removing element from a virtual server farm.

FIG. 5A is a block diagram of an embodiment of a virtual server farm system, computing grid, and supervisory mechanism.

FIG. 5B is a block diagram of a system in which a Supervisor or Control Plane server farm is protected by a firewall.

FIG. 6 is a block diagram of logical connections of a virtual server farm.

FIG. 7 is a block diagram of logical connections of a virtual server farm.

FIG. 8 is a block diagram of logical connections of a virtual server farm.

FIG. 9 is a block diagram of a Control Plane server farm.

FIG. 10 is a block diagram showing connections of Control Plane machines to an embodiment that uses a plurality of SAN switches ("SAN mesh").

FIG. 11 is a block diagram of a plurality of VSFs extended over WAN connections.

FIG. 12 is a block diagram of a computer system with which an embodiment may be implemented.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

A method and apparatus for providing an extensible computing system is described. In the following description, for the purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, to one skilled in the art that the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form in order to avoid unnecessarily obscuring the present invention.

VIRTUAL SERVER FARM (VSF)

According to one embodiment, a wide scale computing fabric ("computing grid") is provided. The computing grid can be physically constructed once, and then logically divided up for various organizations on demand. A part of the computing grid is allocated to each of a plurality of enterprises or organizations. Each organization's logical portion of the computing grid is referred to as a Virtual Server Farm (VSF). Each organization retains independent administrative control of its VSF. Each VSF can change dynamically in terms of number of CPUs, storage capacity and disk and network bandwidth based on real-time demands placed on the server farm or other factors. Each VSF is secure from every other organizations' VSF, even though they are all logically created out of the same physical computing grid. A VSF can be connected back to an Intranet using either a private leased line or a Virtual Private Network (VPN), without exposing the Intranet to other organizations' VSFs.

An organization can access only the data and computing elements in the portion of the computing grid allocated to it, that is, in its VSF, even though it may exercise full (e.g. super-user or root) administrative access to these computers and can observe all traffic on Local Area Networks (LANs) to which these computers are connected. This is

accomplished using a dynamic fire-walling scheme, where the security perimeter of the VSF expands and shrinks dynamically.

Each VSF can be used to host the content and applications of an organization which may be accessed via the Internet, Intranet or Extranet.

Configuration and control of the computing elements and their associated networking and storage elements is performed by a supervisory mechanism which is not directly accessible through any of the computing elements in the computing grid. For convenience, in this document the supervisory mechanism is called Control Plane and may comprise one or more processors or a network of processors. The supervisory mechanism may comprise a Supervisor, Controller, etc. Other approaches may be used, as described herein.

The Control Plane runs on a completely independent set of computing elements assigned for supervisory purposes, such as one or more servers that may be interconnected in a network or by other means. It performs control actions on the computing, networking and storage elements of the computing grid through special control ports or interfaces of the networking and storage elements in the grid. The Control Plane provides a physical interface to switching elements of the system, monitors loads of computing elements in the system, and provides administrative and management functions using a graphical user interface or other suitable user interface.

Computers running the Control Plane are logically invisible to the computers in the computing grid (and therefore in any specific VSF) and cannot be attacked or subverted in any way via elements in the computing grid or from external computers. Only the Control Plane has physical connections to the control ports on devices in the computing grid, which controls membership in a particular VSF. The devices in the computing can be configured only through these special control ports, and therefore computing elements in the computing grid are unable to change their security perimeter or access storage or computing devices which they are not authorized to do.

Thus, a VSF allows organizations to work with computing facilities that appear to comprise a private server farm, dynamically created out of a large-scale shared computing

infrastructure, namely the computing grid. A Control Plane coupled with the computing architecture described herein provides a private server farm whose privacy and integrity is protected through access control mechanisms implemented in the hardware of the devices of the computing grid.

The internal topology of each VSF is controlled by the Control Plane. The Control Plane can take the basic interconnection of computers, network switches and storage network switches described herein and use them to create a variety of server farm configurations. These include but are not limited to, single-tier Web server farms front-ended by a load balancer, as well as multi-tier configurations, where a Web server talks to an application server, which in turn talks to a database server. A variety of load balancing, multi-tiering and fire-walling configurations are possible.

THE COMPUTING GRID

The computing grid may exist in a single location or may be distributed over a wide area. First this document describes the computing grid in the context of a single building-sized network, composed purely of local area technologies. Then the document describes the case where the computing grid is distributed over a wide area network (WAN).

FIG. 2 is a block diagram of one configuration of an extensible computing system 200 that includes a local computing grid 208. In this document "extensible" generally means that the system is flexible and scalable, having the capability to provide increased or decreased computing power to a particular enterprise or user upon demand. The local computing grid 208 is composed of a large number of computing elements CPU1, CPU2, ... CPUn. In an exemplary embodiment, there may be 10,000 computing elements, or more. These computing elements do not contain or store any long-lived per-element state information, and therefore may be configured without persistent or non-volatile storage such as a local disk. Instead, all long lived state information is stored separate from the computing elements, on disks DISK1, DISK2, ... DISKn that are coupled to the computing elements via a Storage Area Network (SAN) comprising one or more SAN

Switches 202. Suitable SAN switches are commercially available from Brocade and Excel.

All of the computing elements are interconnected to each other through one or more VLAN switches 204 which can be divided up into Virtual LANs (VLANs). The VLAN switches 204 are coupled to the Internet 106. In general a computing element contains one or two network interfaces connected to the VLAN switch. For the sake of simplicity, in FIG. 2 all nodes are shown with two network interfaces, although some may have less or more network interfaces. Many commercial vendors now provide switches supporting VLAN functionality. For example, suitable VLAN switches are commercially available from Cisco Systems, Inc. and Xtreme Networks. Similarly there are a large number of commercially available products to construct SANs, including Fibre Channel switches, SCSI-to-Fibre-Channel bridging devices, and Network Attached Storage (NAS) devices.

Control Plane 206 is coupled by a SAN Control path, CPU Control path, and VLAN Control path to SAN switches 202, CPUs CPU1, CPU2, ... CPUn, and VLAN Switches 204, respectively.

Each VSF is composed of a set of VLANs, a set of computing elements that are attached to the VLANs, and a subset of the storage available on the SAN that is coupled to the set of computing elements. The subset of the storage available on the SAN is referred to as a SAN Zone and is protected by the SAN hardware from access from computing elements which are part of other SAN zones. Preferably, VLANs that provide non-forgeable port identifiers are used to prevent one customer or end user from obtaining access to VSF resources of another customer or end user.

FIG. 3 is a block diagram of an exemplary virtual server farm featuring a SAN Zone. A plurality of Web servers WS1, WS2, etc. are coupled by a first VLAN (VLAN1) to a load balancer(LB)/firewall 302. A second VLAN (VLAN2) couples the Internet 106 to the load balancer(LB)/firewall 302. Each of the Web servers may be selected from among CPU1, CPU2, etc., using mechanisms described further herein. The Web servers

are coupled to a SAN Zone 304, which is coupled to one or more storage devices 306a, 306b.

At any given point in time, a computing element in the computing grid, such as CPU1 of FIG. 2, is only connected to the set of VLANs and the SAN zone(s) associated with a single VSF. A VSF typically is not shared among different organizations. The subset of storage on the SAN which belongs to a single SAN zone, and the set of VLANs associated with it and the computing elements on these VLANs define a VSF.

By controlling the membership of a VLAN and the membership of a SAN zone, Control Planc enforces a logical partitioning of the computing grid into multiple VSFs. Members of one VSF cannot access the computing or storage resources of another VSF. Such access restrictions are enforced at the hardware level by the VLAN switches, and by port-level access control mechanisms (e.g., zoning) of SAN hardware such as Fibre Channel switches and edge devices such as SCSI to Fibre Channel bridging hardware. Computing elements that form part of the computing grid are not physically connected to the control ports or interfaces of the VLAN switches and the SAN switches, and therefore cannot control the membership of the VLANs or SAN zones. Accordingly, the computing elements of the computing grid cannot access computing elcmcnts not located in the VSF in which they are contained.

Only the computing elements that run the Control Plane are physically connected to the control ports or interface of the devices in the grid. Devices in the computing grid (computers, SAN switches and VLAN switches) can only be configured through such control ports or interfaces. This provides a simple yet highly secure means of enforcing the dynamic partitioning of the computing grid into multiple VSFs.

Each computing element in a VSF is replaceable by any other computing element. The number of computing elements, VLANs and SAN zones associated with a given VSF may change over time under control of the Control Plane.

In one embodiment, the computing grid includes an Idle Pool that comprises large number of computing elements that are kept in reserve. Computing elements from the Idle Pool may be assigned to a particular VSF for reasons such as increasing the CPU or

memory capacity available to that VSF, or to deal with failures of a particular computing element in a VSF. When the computing elements are configured as Web servers, the Idle Pool serves as a large "shock absorber" for varying or "bursty" Web traffic loads and related peak processing loads.

The Idle Pool is shared between many different organizations, and therefore it provides economies of scale, since no single organization has to pay for the entire cost of the Idle Pool. Different organizations can obtain computing elements from the Idle Pool at different times in the day, as needed, thereby enabling each VSF to grow when required and shrink when traffic falls down to normal. If many different organizations continue to peak at the same time and thereby potentially exhaust the capacity of the Idle Pool, the Idle Pool can be increased by adding more CPUs and storage elements to it (scalability). The capacity of the Idle Pool is engineered so as to greatly reduce the probability that, in steady state, a particular VSF may not be able to obtain an additional computing element from the Idle Pool when it needs to.

FIG. 4A, FIG. 4B, FIG. 4C, and FIG. 4D are block diagrams showing successive steps involved in moving a computing element in and out of the Idle Pool. Referring first to FIG. 4A, assume that the Control Plane has logically connected elements of the computing grid into first and second VSFs labeled VSF1, VSF2. Idle Pool 400 comprises a plurality of CPUs 402, one of which is labeled CPUX. In FIG. 4B, VSF1 has developed a need for an additional computing element. Accordingly, the Control Plane moves CPUX from Idle Pool 400 to VSF1, as indicated by path 404.

In FIG. 4C, VSF1 no longer needs CPUX, and therefore the Control Plane moves CPUX out of VSF1 and back into the Idle Pool 400. In FIG. 4D, VSF2 has developed a need for an additional computing element. Accordingly, the Control Plane moves CPUX from the Idle Pool 400 to VSF2. Thus, over the course of time, as traffic conditions change, a single computing element may belong to the Idle Pool (FIG. 4A), then be assigned to a particular VSF (FIG. 4B), then be placed back in the Idle Pool (FIG. 4C), and then belong to another VSF (FIG. 4D).

At each one of these stages, The Control Plane configures the LAN switches and SAN switches associated with that computing element to be part of the VLANs and SAN zones associated with a particular VSF (or the Idle Pool). According to one embodiment, in between each transition, the computing element is powered down or rebooted. When it is powered back up, it views a different portion of storage zone on the SAN, which includes a bootable image of an operating system (e.g., Linux, NT, Solaris, etc.). The storage zone also includes a data portion that is specific to each organization (e.g., files associated with a Web server, database partitions, etc.). It is also part of another VLAN which is part of the VLAN set of another VSF, so it can access CPUs, SAN storage devices and NAS devices associated with the VLANs of the VSF into which it has been transitioned.

In a preferred embodiment, the storage zones include a plurality of pre-defined logical blueprints that are associated with roles that may be assumed by the computing elements. Initially, no computing element is dedicated to any particular role or task such as Web server, application server, database server, etc. The role of the computing element is acquired from one of a plurality of pre-defined, stored blueprints, each of which defines a boot image for the computing elements that are associated with that role. The blueprints may be stored in the form of a file, a database table, or any other storage format that can associate a boot image location with a role.

Thus, the movements of CPUX in FIG. 4A, FIG. 4B, FIG. 4C, FIG. 4D are logical, not physical, and are accomplished by re-configuring VLAN switches and SAN Zones under control of The Control Plane. Further, each computing element in the computing grid initially is essentially fungible, and assumes a specific processing role only after it is connected in a virtual server farm and loads software from a boot image. No computing element is dedicated to any particular role or task such as Web server, application server, database server, etc. The role of the computing element is acquired from one of a plurality of pre-defined, stored blueprints, each of which is associated with a role, each of which defines a boot image for the computing elements that are associated with that role.

Since there is no long-lived state information stored in any given computing element (such as a local disk), nodes are easily moved between different VSFs, and can run completely different OS and application software. This also makes each computing element highly replaceable, in case of planned or unplanned downtime.

A particular computing element may perform different roles as it is brought into and out of various VSFs. For example, a computing element may act as a Web server in one VSF, and when it is brought into a different VSF, it may be a database server, a Web load balancer, a Firewall, etc. It may also successively boot and run different operating systems such as Linux, NT or Solaris in different VSFs. Thus, each computing element in the computing grid is fungible, and has no static role assigned to it. Accordingly, the entire reserve capacity of the computing grid can be used to provide any of the services required by any VSF. This provides a high degree of availability and reliability to the services provided by a single VSF, because each server performing a particular service has potentially thousands of back-up servers able to provide the same service.

Further, the large reserve capacity of the computing grid can provide both dynamic load balancing properties, as well as high processor availability. This capability is enabled by the unique combination of diskless computing elements interconnected via VLANs, and connected to a configurable zone of storage devices via a SAN, all controlled in real-time by The Control Plane. Every computing element can act in the role of any required server in any VSF, and can connect to any logical partition of any disk in the SAN. When the grid requires more computing power or disk capacity, computing elements or disk storage is manually added to the idle pool, which may decrease over time as more organizations are provided VSF services. No manual intervention is required in order to increase the number of CPUs, network and disk bandwidth and storage available to a VSF. All such resources are allocated on demand from CPU, network and disk resources available in the Idle Pool by the Control Plane.

A particular VSF is not subjected to manual reconfiguration. Only the machines in the idle pool are manually configured into the computing grid. As a result, a great potential hazard present in current manually constructed server farms is removed. The

possibility that human error in configuring a new server into a live server farm can cause the server farm to malfunction, possibly resulting in loss of service to users of that Web site, is virtually eliminated.

The Control Plane also replicates data stored in SAN attached storage devices, so that failure of any particular storage element does not cause a loss of service to any part of the system. By decoupling long-lived storage from computing devices using SANs, and by providing redundant storage and computing elements, where any computing element can be attached to any storage partition, a high degree of availability is achieved.

A DETAILED EXAMPLE OF ESTABLISHING A VIRTUAL SERVER FARM, ADDING A PROCESSOR TO IT, AND REMOVING A PROCESSOR FROM IT

FIG. 5A is a block diagram of a VSF system according to an embodiment. With reference to FIG. 5A, the following describes the detailed steps that may be used to create a VSF, add nodes to it and delete nodes from it.

FIG. 5A depicts computing elements 502, comprising computers A through G, coupled to VLAN capable switch 504. VLAN switch 504 is coupled to Internet 106, and the VLAN switch has ports V1, V2, etc. Computers A through G are further coupled to SAN switch 506, which is coupled to a plurality of storage devices or disks D1-D5. The SAN switch 506 has ports S1, S2, etc. A Control Plane machine 508 is coupled by control paths and data paths to SAN switch 506 and to VLAN switch 504. The Control Plane is able to send control commands to these devices through the control ports.

For the sake of simplicity and exposition, the number of computing elements in FIG. 5A is a small number. In practice, a large number of computers, e.g., thousands or more, and an equally large number of storage devices form the computing grid. In such larger structures, multiple SAN switches are interconnected to form a mesh, and multiple VLAN switches are interconnected to form a VLAN mesh. For clarity and simplicity, however, FIG. 5A shows a single SAN switch and a single VLAN switch.

Initially, all computers A-G are assigned to the idle pool until the Control Plane receives a request to create a VSF. All ports of the VLAN switch are assigned to a

specific VLAN which we shall label as VLAN 1 (for the idle zone). Assume that the Control Plane is asked to construct a VSF, containing one load balancer/firewall and two Web servers connected to a storage device on the SAN. Requests to Control Plane may arrive through a management interface or other computing element.

In response, the Control Plane assigns or allocates CPU A as the load balancer/firewall, and allocates CPUs B and C as the Web servers. CPU A is logically placed in SAN Zone 1, and pointed to a bootable partition on a disk that contains dedicated load balancing/firewalling software. The term "pointed to" is used for convenience and is intended to indicate that CPU A is given, by any means, information sufficient to enable CPU A to obtain or locate appropriate software that it needs to operate. Placement of CPU A in SAN Zone 1 enables CPU A to obtain resources from disks that are controlled by the SAN of that SAN Zone.

The load balancer is configured by the Control Plane to know about CPUs B and C as the two Web servers it is supposed to load balance. The firewall configuration protects CPUs B and C against unauthorized access from the Internet 106. CPUs B and C are pointed to a disk partition on the SAN that contains a bootable OS image for a particular operating system (e.g., Solaris, Linux, NT etc) and Web server application software (e.g., Apache). The VLAN switch is configured to place ports v1 and v2 on VLAN 1, and ports v3, v4, v5, v6 and v7 on VLAN 2. The Control Plane configures the SAN switch 506 to place Fibre-Channel switch ports s2, s3 and s8 into SAN zone 1.

A description of how a CPU is pointed to a particular disk drive, and what this means for booting up and shared access to disk data, is provided further herein.

FIG. 6 is a block diagram of the resulting the logical connectivity of computing elements, which are collectively called VSF 1. Disk drive DD1 is selected from among storage devices D1, D2, etc. Once the logical structure as shown in FIG. 6 is achieved, CPUs A, B, C are given a power-up command. In response, CPU A becomes a dedicated load balancer/firewall machine, and CPUs B, C become Web servers.

Now, assume that because of a policy based rule, the Control Plane determines that another Web server is required in VSF 1. This could be because there is an increased

number of requests coming to the Web site, and the customers plan permits at least three Web servers to be added to VSF 1. Or it may be because the organization that owns or operates the VSF wants another server, and has added it through an administrative mechanism, such as a privileged Web page which allows it to add more servers to its VSF.

In response, the Control Plane decides to add CPU D to VSF 1. In order to do this, the Control Plane will add CPU D to VLAN 2 by adding ports v8 and v9 to VLAN 2. Also, CPU D's SAN port s4 is added to SAN zone 1. CPU D is pointed to a bootable portion of the SAN storage which boots up and runs as a Web server. CPU D also gets read-only access to the shared data on the SAN, which may consist of Web page contents, executable server scripts, etc. This way it is able to serve Web requests intended for the server farm much as CPUs B and C serve requests. The Control Plane will also configure the load balancer (CPU A) to include CPU D as part of the server set which is being load balanced.

CPU D is now booted up, and the size of the VSF has now increased to three Web servers and 1 load balancer. FIG. 7 is a block diagram of the resulting logical connectivity.

Assume that the Control Plane now receives a request to create another VSF, which it will name VSF 2, and which needs two Web servers and one load balancer/firewall. The Control Plane allocates CPU E to be the load balancer/firewall and CPUs F, G to be the Web servers. It configures CPU E to know about CPUs F, G as the two machines to load balance against.

To implement this configuration, the Control Plane will configure VLAN switch 504 to include port v10, v11 in VLAN 1 (that is, connected to the Internet 106) and ports v12, v13 and v14, v15 to be in VLAN 3. Similarly, it configures SAN switch 506 to include SAN ports s6 and s7 and s9 in SAN zone 2. This SAN zone includes the storage containing the software necessary to run CPU E as a load-balancer and CPUs F and G as Web servers that use a shared read-only disk partition contained in Disk D2 in SAN zone 2.

FIG. 8 is a block diagram of the resulting logical connectivity. Although two VSFs (VSF 1, VSF 2) share the same physical VLAN switch and SAN switch, the two VSFs are logically partitioned. Users who access CPUs B, C, D, or the enterprise that owns or operates VSF 1 can only access the CPUs and storage of VSF 1. Such users cannot access the CPUs or storage of VSF 2. This occurs because of the combination of the separate VLANs and the 2 firewalls on the only shared segment (VLAN 1), and the different SAN zones in which the two VSFs are configured.

Further assume that later, the Control Plane decides that VSF 1 can now fall back down to two Web servers. This may be because the temporary increase in load on VSF 1 has decreased, or it may be because of some other administrative action taken. In response, the Control Plane will shut down CPU D by a special command which may include powering down the CPU. Once the CPU has shut down, the Control Plane removes ports v8 and v9 from VLAN 2, and also removes SAN port s4 from SAN zone 1. Port s4 is placed in an idle SAN zone. The idle SAN zone may be designated, for example, SAN Zone I (for Idle) or Zone 0.

Some time later, the Control Plane may decide to add another node to VSF 2. This may be because the load on the Web servers in VSF 2 has temporarily increased or it may be due to other reasons. Accordingly, the Control Plane decides to place CPU D in VSF 2, as indicated by dashed path 802. In order to do this, it configures the VLAN switch to include ports v8, v9 in VLAN 3 and SAN port s4 in SAN zone 2. CPU D is pointed to the portion of the storage on disk device 2 that contains a bootable image of the OS and Web server software required for servers in VSF 2. Also, CPU D is granted read-only access to data in a file system shared by the other Web servers in VSF 2. CPU D is powered back up, and it now runs as a load-balanced Web server in VSF 2, and can no longer access any data in SAN zone 1 or the CPUs attached to VLAN 2. In particular, CPU D has no way of accessing any element of VSF 1, even though at an earlier point in time it was part of VSF 1.

Further, in this configuration, the security perimeter enforced by CPU E has dynamically expanded to include CPU D. Thus, embodiments provide dynamic

firewalling that automatically adjusts to properly protect computing elements that are added to or removed from a VSF.

DISK DEVICES ON THE SAN

There are several ways by which a CPU can be pointed to a particular device on the SAN, for booting up purposes, or for accessing disk storage which needs to be shared with other nodes, or otherwise provided with information about where to find bootup programs and data.

One way is to use a SCSI-to-Fibre Channel bridging device attached to a computing element and provide a SCSI interface for the local disks. By routing that SCSI port to the right drive on the Fibre-Channel SAN, the computer can access the storage device on the Fibre-Channel SAN just as it would access a locally attached SCSI disk. Therefore, software such as boot-up software simply boots off the disk device on the SAN just as it would boot off a locally attached SCSI disk.

Another way is to have a Fibre-Channel interface on the node and associated device-driver and boot ROM and OS software that permits the Fibre-Channel interface to be used as a boot device.

Yet another way is to have an interface card (e.g. PCI bus or Sbus) which appears to be a SCSI or IDE device controller but that in turn communicates over the SAN to access the disk. Operating systems such as Solaris and Windows NT integrally provide diskless boot functions that can be used in this alternative.

Typically there will be two kinds of SAN disk devices associated with a given node. The first is one which is not logically shared with other computing elements, and constitutes what is normally a per-node root partition containing bootable OS images, local configuration files, etc. This is the equivalent of the root filesystem on a Unix system.

The second kind of disk is shared storage with other nodes. The kind of sharing varies by the OS software running on the CPU and the needs of the nodes accessing the shared storage. If the OS provides a cluster file system that allows read/write access of a

shared-disk partition between multiple nodes, the shared disk is mounted as such a cluster file system. Similarly, the system may use database software such as Oracle Parallel Server that permits multiple nodes running in a cluster to have concurrent read/write access to a shared disk. In such cases, a shared disk is already designed into the base OS and application software.

For operating systems where such shared access is not possible, because the OS and associated applications cannot manage a disk device shared with other nodes, the shared disk can be mounted as a read-only device. For many Web applications, having read-only access to Web related files is sufficient. For example, in Unix systems, a particular filesystem may be mounted as read-only.

MULTI-SWITCH COMPUTING GRID

The configuration described above in connection with FIG. 5A can be expanded to a large number of computing and storage nodes by interconnecting a plurality of VLAN switches to form a large switched VLAN fabric, and by interconnecting multiple SAN switches to form a large switched SAN mesh. In this case, a computing grid has the architecture generally shown in FIG. 4, except that the SAN/VLAN switched mesh contains a very large number of ports for CPUs and storage devices. A number of machines running the Control Plane can be physically connected to the control ports of the VLAN/SAN switches, as described further below. Interconnection of multiple VLAN switches to create complex multi-campus data networks is known in this field. See, for example, G. Haviland, "Designing High-Performance Campus Intranets with Multilayer Switching," Cisco Systems, Inc., available online at http://www.cisco.com/warp/public/cc/sol/mkt/ent/ndsgn/highd_wp.htm

SAN ARCHITECTURE

The description assumes that the SAN comprises Fibre-Channel switches and disk devices, and potentially Fibre-Channel edge devices such as SCSI-to-Fibre Channel bridges. However, SANs may be constructed using alternative technologies, such as

Gigabit Ethernet switches, or switches that use other physical layer protocols. In particular, there are efforts currently underway to construct SANs over IP networks by running the SCSI protocol over IP. The methods and architecture described above is adaptable to these alternative methods of constructing a SAN. When a SAN is constructed by running a protocol like SCSI over IP over a VLAN capable layer 2 environment, then SAN zones are created by mapping them to different VLANs.

Also, Network Attached Storage (NAS) may be used, which works over LAN technologies such as fast Ethernet or Gigabit Ethernet. With this option, different VLANs are used in place of the SAN zones in order to enforce security and the logical partitioning of the computing grid. Such NAS devices typically support network filesystems such as Sun's NSF protocol, or Microsoft's SMB, to allow multiple nodes to share the same storage.

CONTROL PLANE IMPLEMENTATION

In the descriptions above, the Control Plane is represented as a box coupled to control and data ports of the SAN/VLAN switches. However, other implementations of Control Plane are contemplated.

Typically the SAN/VLAN control ports are Ethernet interfaces. FIG. 9 is a block diagram of an architecture that can be used in such a case. All the control ("CTL") ports of each VLAN switch (VLAN SW1, VLAN SWn) and all the control ports of each SAN switch (SAN SW1, SAN SWn) are placed on a single Ethernet subnet 902. The subnet 902 is connected only to a plurality of Control Plane machines CP CPU1, CP CPU2, etc. This permits multiple Control Plane machines to be connected to the control ports of all the SAN switches and VLAN switches.

In this configuration, the plurality of Control Plane machines are referred to collectively as the Control Plane or CP 904. Only machines in CP 904 have a physical connection to the control ports of the VLAN switches and SAN switches. Accordingly, CPUs in a given VSF cannot change the membership of the VLANs and SAN zones associated with its own VSF or any other VSF.

Alternatively, instead of Ethernet interfaces, the control ports could be serial or parallel ports. In this case, the ports are coupled to the Control Plane machines.

CONTROL PLANE DATA CONNECTIONS TO VLANS

The machines running the Control Plane need to have access to the data ports on both the VLAN switches as well as the SAN switches. This is required in order for the Control Plane to be able to configure the files related to a particular node and to collect real-time information from a node related to current CPU load, network load, and disk load.

FIG. 5B is a block diagram of one embodiment showing a configuration for connecting a Control Plane 516 to data ports. In one embodiment, machines in each VSF periodically send a packet to a machine 510 that is acting as an agent for the Control Plane. Alternatively, the Control Plane agent machine 510 can periodically poll the nodes in the VSF for their real-time data. The Control Plane agent machine 510 then sends the data it collects from all the nodes in a VSF to the CP 516. Each machine in CP 516 is coupled to a CP LAN 514. The CP LAN 514 is coupled to a special port V17 of VLAN Switch 504 through a CP firewall 512. This provides a scalable and secure means for the CP to collect all the real-time information from the nodes in all the VSFs.

CONTROL PLANE TO SAN DATA CONNECTIONS

FIG. 10 is a block diagram showing connections of Control Plane machines to an embodiment that uses a plurality of SAN switches ("SAN mesh"). A plurality of Control Plane machines CP CPU1, CP CPU2, etc., form a Control Plane Server Farm (CP) 904. Each Control Plane machine is coupled to a port of the SAN mesh.

Associated with the Control Plane machines is a set of SAN ports So, Sp that are connected to disks 1004 that contain data private to the Control Plane. Disks 1004 are logically located in Control Plane Private Storage Zone 1002, which is an area where the Control Plane maintains log files, statistical data, current Control Plane configuration information, and software that implements the Control Plane. SAN ports So, Sp are only

part of the Control Plane SAN zone. Ports So, Sp are never placed on any other SAN zone, and only machines which are part of the CP 904 can access the disks 1004 connected to these ports.

Ports S1, S2 and Sn and ports So and Sp are in the Control Plane SAN zone. No computing element from either the Idle Pool or any VSF is part of the Control Plane SAN zone. This ensures that Control Plane private data is protected from access from any VSF.

When a particular Control Plane machine needs to access a disk partition which is part of a particular VSF, such as VSF I of FIG. 10, then it is placed in the SAN zone associated with that VSF. In this example, CP CPU 2 needs to access the disks of VSF I, so port s2 which is associated with CP CPU 2 is placed in the SAN zone of VSF I, which includes port si. Once the CP CPU is done accessing the disks on port si, it is removed from the SAN zone of VSF I.

Similarly, if a machine such as CP CPU 1 needs to access the disks of VSF j, then it is placed in the SAN zone associated with VSF j. As a result, port s2 is placed in the SAN zone associated with VSF j, which includes the zone containing port sj. Once CP 1 is done accessing the disks connected to port sj, it is removed from the SAN zone associated with VSF j.

CONTROL PLANE TO VLAN DATA CONNECTIONS

Control Plane machines need to collect information from the computing nodes, such as real-time load related information. In order to do this Control Plane needs to have network connectivity to the nodes in the grid itself.

A WIDE AREA COMPUTING GRID

The VSF described above can be distributed over a WAN in several ways.

In one alternative, a wide area backbone may be based on Asynchronous Transfer Mode (ATM) switching. In this case, each local area VLAN is extended into a wide area using Emulated LANs (ELANs) which are part of the ATM LAN Emulation (LANE) standard. In this way, a single VSF can span across several wide area links, such as

ATM/SONET/OC-12 links. An ELAN becomes part of a VLAN which extends across the ATM WAN.

Alternatively, a VSF is extended across a WAN using a VPN system. In this embodiment, the underlying characteristics of the network become irrelevant, and the VPN is used to interconnect two or more VSFs across the WAN to make a single distributed VSF.

Data mirroring technologies can be used in order to have local copies of the data in a distributed VSF. Alternatively, the SAN is bridged over the WAN using one of several SAN to WAN bridging techniques, such as SAN-to-ATM bridging or SAN-to-Gigabit Ethernet bridging. SANs constructed over IP networks naturally extend over the WAN since IP works well over such networks.

FIG. 11 is a block diagram of a plurality of VSFs extended over WAN connections. A San Jose Center, New York Center, and London center are coupled by WAN connections. Each WAN connection comprises an ATM, ELAN, or VPN connection in the manner described above. Each center comprises at least one VSF and at least one Idle Pool. For example, the San Jose center has VSF1A and Idle Pool A. In this configuration, the computing resources of each Idle Pool of a center are available for allocation or assignment to a VSF located in any other center. When such allocation or assignment is carried out, a VSF becomes extended over the WAN.

EXAMPLE USES OF VSFS

The VSF architecture described in the examples above may be used in the context of Web server system. Thus, the foregoing examples have been described in terms of Web servers, application servers and database servers constructed out of the CPUs in a particular VSF. However, the VSF architecture may be used in many other computing contexts and to provide other kinds of services; it is not limited to Web server systems.

-- A DISTRIBUTED VSF AS PART OF A CONTENT DISTRIBUTION NETWORK

In one embodiment, a VSF provides a Content Distribution Network (CDN) using a wide area VSF.

The CDN is a network of caching servers that performs distributed caching of data. The network of caching servers may be implemented, for example, using TrafficServer (TS) software commercially available from Inktomi Corporation, San Mateo, California. TS is a cluster aware system; the system scales as more CPUs are added to a set of caching Traffic Server machines. Accordingly, it is well suited to a system in which adding CPUs is the mechanism for scaling upwards.

In this configuration, a system can dynamically add more CPUs to that portion of a VSF that runs caching software such as TS, thereby growing the cache capacity at a point close to where bursty Web traffic is occurring. As a result, a CDN may be constructed that dynamically scales in CPU and I/O bandwidth in an adaptive way.

-- A VSF FOR HOSTED INTRANET APPLICATIONS

There is growing interest in offering Intranet applications such as Enterprise Resource Planning (ERP), ORM and CRM software as hosted and managed services. Technologies such as Citrix WinFrame and Citrix MetaFrame allow an enterprise to provide Microsoft Windows applications as a service on a thin client such as a Windows CE device or Web browser. A VSF can host such applications in a scalable manner.

For example, the SAP R/3 ERP software, commercially available from SAP Aktiengesellschaft of Germany, allows an enterprise to load balance using multiple Application and Database Servers. In the case of a VSF, an enterprise would dynamically add more Application Servers (e.g., SAP Dialog Servers) to a VSF in order to scale up the VSF based on real-time demand or other factors.

Similarly, Citrix Metaframe allows an enterprise to scale up Windows application users on a server farm running the hosted Windows applications by adding more Citrix servers. In this case, for a VSF, the Citrix MetaFrame VSF would dynamically add more

Citrix servers in order to accommodate more users of Metaframe hosted Windows applications.

It will be apparent that many other applications may be hosted in a manner similar to the illustrative examples described above.

-- CUSTOMER INTERACTION WITH A VSF

Since a VSF is created on demand, a VSF customer or organization that "owns" the VSF may interact with the system in various ways in order to customize a VSF. For example, because a VSF is created and modified instantly via the Control Plane, the VSF customer may be granted privileged access to create and modify its VSF itself. The privileged access may be provided using password authentication provided by Web pages and security applications, token card authentication, Kerberos exchange, or other appropriate security elements.

In one exemplary embodiment, a set of Web pages are served by the Control Plane machine, or by a separate server. The Web pages enable a customer to create a custom VSF, by specifying a number of tiers, the number of computing elements in a particular tier, the hardware and software platform used for each element, and things such as what kind of Web server, application server, or database server software should be pre-configured on these computing elements. Thus, the customer is provided with a virtual provisioning console.

After the customer or user enters such provisioning information, the Control Plane parses and evaluates the order and queues it for execution. Orders may be reviewed by human managers to ensure that they are appropriate. Credit checks of the enterprise may be run to ensure that it has appropriate credit to pay for the requested services. If the provisioning order is approved, the Control Plane may configure a VSF that matches the order, and return to the customer a password providing root access to one or more of the computing elements in the VSF. The customer may then upload master copies of applications to execute in the VSF.

When the enterprise that hosts the computing grid is a for-profit enterprise, the Web pages may also receive payment related information, such as a credit card, a PO number, electronic check, or other payment method.

In another embodiment, the Web pages enable the customer to choose one of several VSF service plans, such as automatic growth and shrinkage of a VSF between a minimum and maximum number of elements, based on real-time load. The customer may have a control value that allows the customer to change parameters such as minimum number of computing elements in a particular tier such as Web servers, or a time period in which the VSF must have a minimal amount of server capacity. The parameters may be linked to billing software that would automatically adjust the customer's bill rate and generate billing log file entries.

Through the privileged access mechanism the customer can obtain reports and monitor real-time information related to usage, load, hits or transactions per second, and adjust the characteristics of a VSF based on the real-time information.

It will be apparent that the foregoing features offer significant advantages over conventional manual approaches to constructing a server farm. In the conventional approaches, a user cannot automatically influence server farm's properties without going through a cumbersome manual procedure of adding servers and configuring the server farm in various ways.

-- BILLING MODELS FOR A VSF

Given the dynamic nature of a VSF, the enterprise that hosts the computing grid and VSFs may bill service fees to customers who own VSFs using a billing model for a VSF which is based on actual usage of the computing elements and storage elements of a VSF. It is not necessary to use a flat fee billing model. The VSF architecture and methods disclosed herein enable a "pay-as-you-go" billing model because the resources of a given VSF are not statically assigned. Accordingly, a particular customer having a highly variable usage load on its server farm could save money because it would not be billed a

rate associated with constant peak server capacity, but rather, a rate that reflects a running average of usage, instantaneous usage, etc.

For example, an enterprise may operate using a billing model that stipulates a flat fee for a minimum number of computing elements, such as 10 servers, and stipulates that when real-time load requires more than 10 elements, then the user is billed at an incremental rate for the extra servers, based on how many extra servers were needed and for the length of time that they are needed.

The units of such bills may reflect the resources that are billed. For example, bills may be expressed in units such as MIPS-hours, CPU-hours, thousands of CPU seconds, etc.

-- A CUSTOMER VISIBLE CONTROL PLANE API

In another alternative, the capacity of a VSF may be controlled by providing the customer with an application programming interface (API) that defines calls to the Control Plane for changing resources. Thus, an application program prepared by the customer could issue calls or requests using the API to ask for more servers, more storage, more bandwidth, etc. This alternative may be used when the customer needs the application program to be aware of the computing grid environment and to take advantage of the capabilities offered by the Control Plane.

Nothing in the above-disclosed architecture requires the customer to modify its application for use with the computing grid. Existing applications continue to work as they do in manually configured server farms. However, an application can take advantage of the dynamism possible in the computing grid, if it has a better understanding of the computing resources it needs based on the real-time load monitoring functions provided by the Control Plane.

An API of the foregoing nature, which enables an application program to change the computing capacity of a server farm, is not possible using existing manual approaches to constructing a server farm.

-- AUTOMATIC UPDATING AND VERSIONING

Using the methods and mechanisms disclosed herein, the Control Plane may carry out automatic updating and versioning of operating system software that is executed in computing elements of a VSF. Thus, the end user or customer is not required to worry about updating the operating system with a new patch, bug fix, etc. The Control Plane can maintain a library of such software elements as they are received and automatically distribute and install them in computing elements of all affected VSFs.

IMPLEMENTATION MECHANISMS

The computing elements and supervisory mechanism may be implemented in several forms. In one embodiment, each computing element is a general purpose digital computer having the elements shown in FIG. 12 except for non-volatile storage device 1210, and the supervisory mechanism is a general purpose digital computer of the type shown in FIG. 12 operating under control of program instructions that implement the processes described herein.

FIG. 12 is a block diagram that illustrates a computer system 1200 upon which an embodiment of the invention may be implemented. Computer system 1200 includes a bus 1202 or other communication mechanism for communicating information, and a processor 1204 coupled with bus 1202 for processing information. Computer system 1200 also includes a main memory 1206, such as a random access memory (RAM) or other dynamic storage device, coupled to bus 1202 for storing information and instructions to be executed by processor 1204. Main memory 1206 also may be used for storing temporary variables or other intermediate information during execution of instructions to be executed by processor 1204. Computer system 1200 further includes a read only memory (ROM) 1208 or other static storage device coupled to bus 1202 for storing static information and instructions for processor 1204. A storage device 1210, such as a magnetic disk or optical disk, is provided and coupled to bus 1202 for storing information and instructions.

Computer system 1200 may be coupled via bus 1202 to a display 1212, such as a cathode ray tube (CRT), for displaying information to a computer user. An input device 1214, including alphanumeric and other keys, is coupled to bus 1202 for communicating information and command selections to processor 1204. Another type of user input device is cursor control 1216, such as a mouse, a trackball, or cursor direction keys for communicating direction information and command selections to processor 1204 and for controlling cursor movement on display 1212. This input device typically has two degrees of freedom in two axes, a first axis (e.g., x) and a second axis (e.g., y), that allows the device to specify positions in a plane.

The invention is related to the use of computer system 1200 for implementing the methods, mechanisms and architecture described herein. According to one embodiment of the invention, such methods and mechanisms are implemented by computer system 1200 in response to processor 1204 executing one or more sequences of one or more instructions contained in main memory 1206. Such instructions may be read into main memory 1206 from another computer-readable medium, such as storage device 1210. Execution of the sequences of instructions contained in main memory 1206 causes processor 1204 to perform the process steps described herein. In alternative embodiments, hard-wired circuitry may be used in place of or in combination with software instructions to implement the invention. Thus, embodiments of the invention are not limited to any specific combination of hardware circuitry and software.

The term "computer-readable medium" as used herein refers to any medium that participates in providing instructions to processor 1204 for execution. Such a medium may take many forms, including but not limited to, non-volatile media, volatile media, and transmission media. Non-volatile media includes, for example, optical or magnetic disks, such as storage device 1210. Volatile media includes dynamic memory, such as main memory 1206. Transmission media includes coaxial cables, copper wire and fiber optics, including the wires that comprise bus 1202. Transmission media can also take the form of acoustic or light waves, such as those generated during radio-wave and infra-red data communications.

Common forms of computer-readable media include, for example, a floppy disk, a flexible disk, hard disk, magnetic tape, or any other magnetic medium, a CD-ROM, any other optical medium, punch cards, paper tape, any other physical medium with patterns of holes, a RAM, a PROM, and EPROM, a FLASH-EPROM, any other memory chip or cartridge, a carrier wave as described hereinafter, or any other medium from which a computer can read.

Various forms of computer readable media may be involved in carrying one or more sequences of one or more instructions to processor 1204 for execution. For example, the instructions may initially be carried on a magnetic disk of a remote computer. The remote computer can load the instructions into its dynamic memory and send the instructions over a telephone line using a modem. A modem local to computer system 1200 can receive the data on the telephone line and use an infra-red transmitter to convert the data to an infra-red signal. An infra-red detector can receive the data carried in the infra-red signal and appropriate circuitry can place the data on bus 1202. Bus 1202 carries the data to main memory 1206, from which processor 1204 retrieves and executes the instructions. The instructions received by main memory 1206 may optionally be stored on storage device 1210 either before or after execution by processor 1204.

Computer system 1200 also includes a communication interface 1218 coupled to bus 1202. Communication interface 1218 provides a two-way data communication coupling to a network link 1220 that is connected to a local network 1222. For example, communication interface 1218 may be an integrated services digital network (ISDN) card or a modem to provide a data communication connection to a corresponding type of telephone line. As another example, communication interface 1218 may be a local area network (LAN) card to provide a data communication connection to a compatible LAN. Wireless links may also be implemented. In any such implementation, communication interface 1218 sends and receives electrical, electromagnetic or optical signals that carry digital data streams representing various types of information.

Network link 1220 typically provides data communication through one or more networks to other data devices. For example, network link 1220 may provide a

connection through local network 1222 to a host computer 1224 or to data equipment operated by an Internet Service Provider (ISP) 1226. ISP 1226 in turn provides data communication services through the world wide packet data communication network now commonly referred to as the "Internet" 1228. Local network 1222 and Internet 1228 both use electrical, electromagnetic or optical signals that carry digital data streams. The signals through the various networks and the signals on network link 1220 and through communication interface 1218, which carry the digital data to and from computer system 1200, are exemplary forms of carrier waves transporting the information.

Computer system 1200 can send messages and receive data, including program code, through the network(s), network link 1220 and communication interface 1218. In the Internet example, a server 1230 might transmit a requested code for an application program through Internet 1228, ISP 1226, local network 1222 and communication interface 1218. In accordance with the invention, one such downloaded application implements the methods and mechanisms described herein.

The received code may be executed by processor 1204 as it is received, and/or stored in storage device 1210, or other non-volatile storage for later execution. In this manner, computer system 1200 may obtain application code in the form of a carrier wave.

ADVANTAGES; SCOPE

The computing grid disclosed herein may be compared conceptually to the public electric power network that is sometimes called the power grid. The power grid provides a scalable means for many parties to obtain power services through a single wide-scale power infrastructure. Similarly, the computing grid disclosed herein provides computing services to many organizations using a single wide-scale computing infrastructure. Using the power grid, power consumers do not independently manage their own personal power equipment. For example, there is no reason for a utility consumer to run a personal power generator at its facility, or in a shared facility and manage its capacity and growth on an individual basis. Instead, the power grid enables the wide-scale distribution of power to vast segments of the population, thereby providing great economies of scale. Similarly,

the computing grid disclosed herein can provide computing services to vast segments of the population using a single wide-scale computing infrastructure.

In the foregoing specification, the invention has been described with reference to specific embodiments thereof. It will, however, be evident that various modifications and changes may be made thereto without departing from the broader spirit and scope of the invention. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense.

CLAIMS

What is claimed is:

1. A data processing method, comprising the steps of:
selecting, from among a set of processors, a subset of the processors;
generating instructions that cause a first switching system to logically couple together each of the processors in the subset of the processors;
selecting, from among a set of storage devices, a subset of the storage devices;
generating instructions that cause a second switching system to logically couple together each of the storage devices in the subset of the storage devices to one another and to the subset of processors.

2. A method as recited in Claim 1, wherein selecting, from among a set of processors, a subset of the processors comprises the steps of selecting subset of central processing units from among a pool of available central processing units.

3. A method as recited in Claim 1,
wherein the step of selecting, from among a set of processors, a subset of the processors comprises the step of selecting subset of central processing units from among a pool of available central processing units, and
wherein each of the central processing units includes first and second network interfaces configured for receiving instructions from a virtual local area network switch and a storage interface configured for connection to the subset of the storage devices through a storage area network switch.

4. A method as recited in Claim 1, wherein the step of generating instructions that cause a first switching system to logically couple together each of the processors in the subset of the processors comprises the step of generating instructions to a virtual local area network switch coupled to the processors that cause the virtual local area network switch to logically couple together the processors in the subset.

5. A method as recited in Claim 1, wherein the step of selecting, from among a set of storage devices, a subset of the storage devices comprises the steps of selecting the subset of storage devices from among a pool of available storage devices.

6. A method as recited in Claim 1,
wherein the step of selecting, from among a set of storage devices, a subset of the storage devices comprises the steps of selecting the subset of storage devices from among a pool of available storage devices; and
wherein each of the storage devices includes a switching interface configured for receiving instructions from a virtual storage area network switch.

7. A method as recited in Claim 1, wherein generating instructions that cause a second switching system to logically couple together each of the storage devices comprises the step of generating instructions to a virtual storage area network switch coupled to the storage devices that cause the virtual storage area network switch to logically couple together the storage devices in the subset.

8. A method as recited in Claim 1, wherein the step of selecting, from among a set of processors, a subset of the processors is carried out by a controller that is coupled to and controls the first switching system, the second switching system, and the set of processors.

9. A method as recited in Claim 1, further comprising the steps of:
creating a first virtual server farm for use in a first data processing operation by selecting, from among the set of processors, a first subset of the processors to process the first data processing operation; generating instructions that cause the first switching system to logically couple together each of the processors in the first subset of the processors in a first virtual local area network; selecting, from among the set of storage devices, a first subset of the storage devices to store information for the first data processing problem; and generating instructions that cause a second switching system to logically couple together each of the storage devices in the first subset

of the storage devices to one another and to the first subset of processors in a first storage area network zone;

creating a second virtual server farm for use in a second data processing operation by selecting, from among the set of processors, a second subset of the processors to process the second data processing operation; generating instructions that cause the first switching system to logically couple together each of the processors in the second subset of the processors in a second virtual local area network; selecting, from among the set of storage devices, a second subset of the storage devices to store information for the second data processing problem; and generating instructions that cause the second switching system to logically couple together each of the storage devices in the second subset of the storage devices to one another and to the second subset of processors in a second storage area network zone;

wherein the instructions securely isolate the first subset of processors from the second subset of processors and from the second subset of storage devices.

10. A method as recited in Claim 1, further comprising the steps of:

selecting, from among the set of processors, an additional processor;

generating instructions that cause the first switching system to logically couple the additional processor to the processors in the subset of the processors.

11. A method as recited in Claim 1, further comprising the steps of:

selecting, from among the subset of processors, a particular processor to be removed from the subset;

generating instructions that cause the first switching system to logically decouple the particular processor from the subset of the processors.

12. A method as recited in Claim 2, further comprising the steps of:

selecting, from among the subset of processors, a particular processor to be removed from the subset;

generating instructions that cause the first switching system to logically decouple the particular processor from the subset of the processors;

logically placing the particular processor in the pool of available processors.

13. A method as recited in Claim 9, further comprising the steps of:

selecting, from among the first subset of processors, a particular processor to be removed from the first subset;

generating instructions that cause the first switching system to logically decouple the particular processor from the first subset of the processors;

generating instructions that cause the first switching system to logically add the particular processor to the second subset of the processors.

14. A method as recited in Claim 1, further comprising the step of initially assigning all the processors to an idle pool of available processors.

15. A method as recited in Claim 1, further comprising the step of dynamically logically adding or removing one or more processors to or from the subset of processors in response to real-time loading experienced by the subset of processors.

16. A method as recited in Claim 1, further comprising the step of dynamically logically adding or removing one or more storage devices to or from the subset of storage devices in response to real-time loading experienced by the subset of storage devices.

17. A method as recited in Claim 1, further comprising coupling a interface of the first switching system to an external network whereby the subset of processors is responsive to requests from the external network.

18. A method as recited in Claim 15, further comprising the steps of:

logically adding an additional processor to the subset of processors in response to real-time loading experienced by the subset of processors;

re-starting the additional processor;

loading into the additional processor, an operating system image obtained from a pre-defined location of one of the storage devices in the subset of storage devices.

19. A method as recited in Claim 1, further comprising the steps of:
logically coupling a load balancer to the processors in the subset of processors;
instructing the load balancer to load balance processing that is carried out by the processors in the subset of processors.

20. A method as recited in Claim 9, further comprising the step of dynamically logically adding an additional processor to the subset of processors in response to real-time loading experienced by the subset of processors, by:
adding to the virtual local area network those interface ports of the first switching system that are associated with the additional processor;
adding a storage area port of the additional processor to the storage area network zone.

21. A method as recited in Claim 9, further comprising the step of dynamically logically removing one processor from the subset of processors in response to real-time loading experienced by the subset of processors, by:
removing from the virtual local area network those interface ports of the first switching system that are associated with the additional processor;
removing a storage area port of the one processor from the storage area network zone.

22. A method as recited in Claim 8, further comprising the step of logically assigning one or more ports of the second switching system, which ports are associated with one of the storage devices that is assigned exclusively to the controller, to a private storage area network zone for use by the controller.

23. A method as recited in Claim 1, further comprising the step of

associating each of the processors in the subset of processors with a stored, pre-defined blueprint that associates one of a plurality of processing roles with a boot image of one of the storage devices in the subset of storage devices;

generating instructions that cause each of the processors in the subset of processors to load and execute the boot image from the storage device associated with the processing role of that processor.

24. A method of data processing comprising the steps of:

creating a first virtual server farm for use in a first data processing operation by selecting, from among a set of processors, a first subset of the processors to process the first data processing operation; generating instructions that cause the first switching system to logically couple together each of the processors in the first subset of the processors in a first virtual local area network; selecting, from among a set of storage devices, a first subset of the storage devices to store information for the first data processing problem; and generating instructions that cause a second switching system to logically couple together each of the storage devices in the first subset of the storage devices to one another and to the first subset of processors in a first storage area network zone;

creating a second virtual server farm for use in a second data processing operation by selecting, from among the set of processors, a second subset of the processors to process the second data processing operation; generating instructions that cause the first switching system to logically couple together each of the processors in the second subset of the processors in a second virtual local area network; selecting, from among the set of storage devices, a second subset of the storage devices to store information for the second data processing problem; and generating instructions that cause the second switching system to logically couple together each of the storage devices in the second subset of the storage devices to one another and to the second subset of processors in a second storage area network zone;

wherein the instructions securely isolate the first subset of processors from the second subset of processors and from the second subset of storage devices.

25. A data processing system, comprising:

a plurality of processors;

a first switching system coupled to the plurality of processors;

a plurality of storage devices;

a second switching system coupled to the plurality of storage devices;

a controller coupled to the first switching system and the second switching system;

means in the controller for selecting, from among the plurality of processors, a subset of the processors;

means in the controller for generating instructions that cause the first switching system to logically couple together each of the processors in the subset of processors;

means in the controller for selecting, from among the plurality of storage devices, a subset of the storage devices; and

means in the controller for generating instructions that cause the second switching system to logically couple together each of the storage devices in the subset of the storage devices to one another and to the subset of processors.

26. A data processing system as recited in Claim 25, wherein the controller further comprises means for selecting, from among a set of processors, a subset of the processors by selecting subset of central processing units from among a pool of available central processing units.

27. A data processing system as recited in Claim 25,

wherein the means for selecting a subset of the processors comprises means for selecting subset of central processing units from among a pool of available central processing units, and

wherein each of the central processing units includes first and second network interfaces configured for receiving instructions from a virtual local area

network switch and a storage interface configured for connection to the subset of the storage devices through a storage area network switch.

28. A data processing system as recited in Claim 25, wherein the means for generating instructions that cause a first switching system to logically couple together each of the processors in the subset of the processors comprises means for generating instructions to a virtual local area network switch coupled to the processors that cause the virtual local area network switch to logically couple together the processors in the subset.

29. A data processing system as recited in Claim 25, wherein the means for selecting a subset of the storage devices comprises means for selecting the subset of storage devices from among a pool of available storage devices.

30. A data processing system as recited in Claim 25,
wherein the means for selecting a subset of the storage devices comprises means for selecting the subset of storage devices from among a pool of available storage devices; and
wherein each of the storage devices includes a switching interface configured for receiving instructions from a virtual storage area network switch.

31. A data processing system as recited in Claim 25, wherein the means for generating instructions that cause a second switching system to logically couple together each of the storage devices comprises means for generating instructions to a virtual storage area network switch coupled to the storage devices that cause the virtual storage area network switch to logically couple together the storage devices in the subset.

32. A data processing system as recited in Claim 25, further comprising:
a first virtual server farm for use in a first data processing operation that is created by selecting, from among the set of processors, a first subset of the processors to process the first data processing operation; generating instructions that cause the first switching system to logically couple

together each of the processors in the first subset of the processors in a first virtual local area network; selecting, from among the set of storage devices, a first subset of the storage devices to store information for the first data processing problem; and generating instructions that cause a second switching system to logically couple together each of the storage devices in the first subset of the storage devices to one another and to the first subset of processors in a first storage area network zone;

a second virtual server farm for use in a second data processing operation that is created by selecting, from among the set of processors, a second subset of the processors to process the second data processing operation; generating instructions that cause the first switching system to logically couple together each of the processors in the second subset of the processors in a second virtual local area network; selecting, from among the set of storage devices, a second subset of the storage devices to store information for the second data processing problem; and generating instructions that cause the second switching system to logically couple together each of the storage devices in the second subset of the storage devices to one another and to the second subset of processors in a second storage area network zone;

wherein the instructions securely isolate the first subset of processors from the second subset of processors and from the second subset of storage devices.

33. A data processing system as recited in Claim 25, further comprising:

means for selecting, from among the set of processors, an additional processor;

means for generating instructions that cause the first switching system to logically couple the additional processor to the processors in the subset of the processors.

34. A data processing system as recited in Claim 25, further comprising:

means for selecting, from among the subset of processors, a particular processor to be removed from the subset;

means for generating instructions that cause the first switching system to logically decouple the particular processor from the subset of the processors.

35. A data processing system as recited in Claim 26, further comprising:
means for selecting, from among the subset of processors, a particular processor to be removed from the subset;
means for generating instructions that cause the first switching system to logically decouple the particular processor from the subset of the processors;
means for logically placing the particular processor in the pool of available processors.

36. A data processing system as recited in Claim 28, further comprising:
means for selecting, from among the first subset of processors, a particular processor to be removed from the first subset;
means for generating instructions that cause the first switching system to logically decouple the particular processor from the first subset of the processors;
means for generating instructions that cause the first switching system to logically add the particular processor to the second subset of the processors.

37. A data processing system as recited in Claim 25, further comprising means for initially assigning all the processors to an idle pool of available processors.

38. A data processing system as recited in Claim 25, further comprising means for dynamically logically adding or removing one or more processors to or from the subset of processors in response to real-time loading experienced by the subset of processors.

39. A data processing system as recited in Claim 25, further comprising means for dynamically logically adding or removing one or more storage devices to or from the subset of storage devices in response to real-time loading experienced by the subset of storage devices.

40. A data processing system as recited in Claim 25, wherein the controller further comprises a load monitor that generates information representing processing load of each of the processors in the subset.

41. A data processing system as recited in Claim 25, wherein the first switching system comprises a virtual local area network switch having non-forgeable port identifiers.

42. A data processing system as recited in Claim 25, wherein the second switching system comprises a storage area network switch, wherein the subset of storage devices is logically organized in a storage area network zone and wherein the storage area network switch permits access to the subset of storage devices only to those processors in the subset of processors.

43. A data processing system as recited in Claim 25, wherein the second switching system comprises a storage area network switch, wherein the subset of storage devices is logically organized in a storage area network zone, and wherein the storage area network switch permits access to the subset of storage devices only to those processors in the subset of processors using one or more Fibre Channel switches.

44. A data processing system as recited in Claim 25, wherein the second switching system comprises a storage area network switch, wherein the subset of storage devices is logically organized in a storage area network zone, and wherein the subset of processors is coupled to an external network through a load balancer or firewall.

45. A data processing system as recited in Claim 25, further comprising a plurality of controllers logically interconnected in a subnetwork that includes control ports of the first switching system and that includes control ports of the second switching system.

46. A data processing system as recited in Claim 45, further comprising an agent controller configured to periodically poll each processor in the subset of processors for receiving real-time information representing then-current processor load, network load, or storage load, and configured to communicate the information to each of the controllers.

47. A data processing system as recited in Claim 45, further comprising the step of logically assigning one or more ports of the second switching system, which ports are associated with one of the storage devices that is assigned exclusively to the controller, to a private storage area network zone for use by the controller.

48. A data processing system as recited in Claim 25, further comprising:

a plurality of stored, pre-defined blueprints, each of which associates one of a plurality of processing roles with a boot image of one of the storage devices in the subset of storage devices;

means for associating each of the processors in the subset of processors with one of the blueprints and for causing each of the processors in the subset of processors to load and execute the boot image from the storage device associated with the processing role of that processor.

49. A virtual computing system, comprising:

a plurality of processors logically organized in a plurality of subsets of the processors, each of the subsets logically organized as one of a plurality of virtual local area networks;

a plurality of storage devices logically organized in a storage area network coupled to the plurality of virtual local area networks for storing data and instructions used by one of the subsets; and

a control plane coupled to the plurality of virtual local area networks and to the storage area network, and configured to dynamically add or remove processors from the subsets and to dynamically add or remove storage devices from the storage area network in response to changing processing load conditions and changing storage needs that occur in real time.

50. A virtual computing system as recited in Claim 49, further comprising:

a plurality of stored, pre-defined blueprints, each of which associates one of a plurality of processing roles with a boot image of one of the storage devices in the subset of storage devices;

means for associating each of the processors one of the subsets of processors with one of the blueprints and for causing each of the processors in that subset

of processors to load and execute the boot image from the storage device associated with the processing role of that processor.

51. A computer-readable medium carrying one or more sequences of instructions for data processing using an extensible computing system, wherein execution of the one or more sequences of instructions by one or more processors causes the one or more processors to perform the steps of:

selecting, from among a set of processors, a subset of the processors;

generating instructions that cause a first switching system to logically couple together each of the processors in the subset of the processors;

selecting, from among a set of storage devices, a subset of the storage devices;

generating instructions that cause a second switching system to logically couple together each of the storage devices in the subset of the storage devices to one another and to the subset of processors.

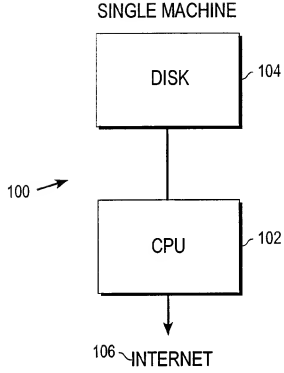

SINGLE MACHINE
DISK
104
100
CPU
102
106
INTERNET

Fig. 1A

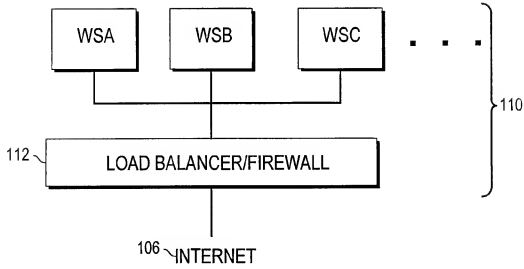

WSA
WSB
WSC
110
112
LOAD BALANCER/FIREWALL
106
INTERNET

Fig. 1B

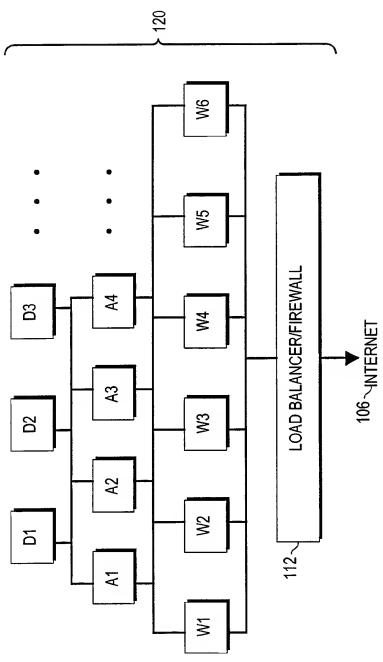

120
D1
D2
D3
A1
A2
A3
A4
W1
W2
W3
W4
W5
W6
LOAD BALANCER/FIREWALL
112
106
INTERNET
DATABASE SERVERS
APPLICATION SERVERS
WEB SERVERS

Fig. 16

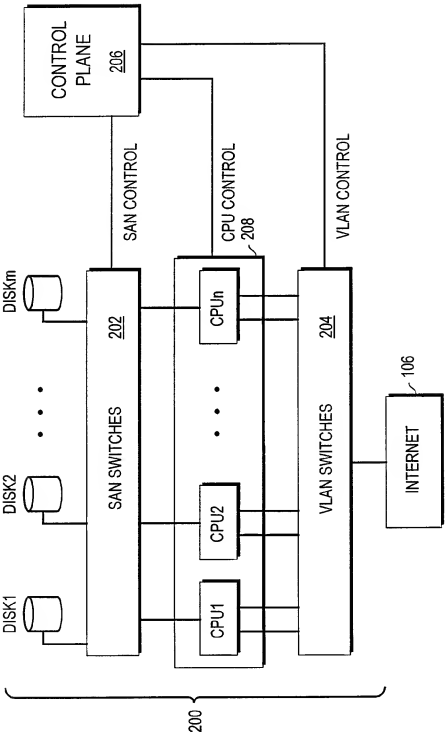

CONTROL PLANE
206
SAN CONTROL
CPU CONTROL
208
VLAN CONTROL
DISKm
DISK2
DISK1
SAN SWITCHES
202
CPUn
CPU2
CPU1
VLAN SWITCHES
204
106
INTERNET
200

Fig. 2

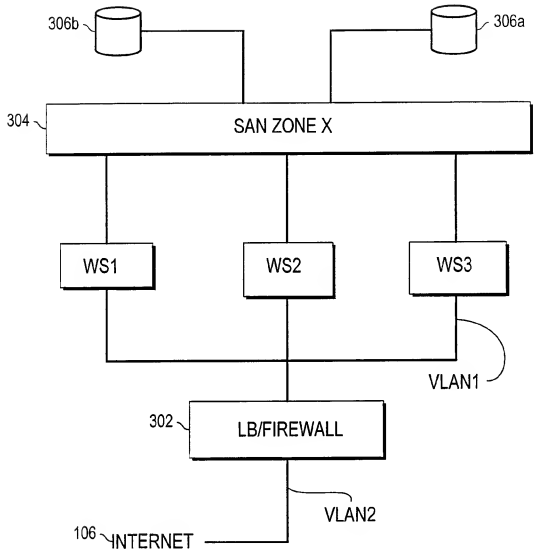

306b
306a
304
SAN ZONE X
WS1
WS2
WS3
VLAN1
302
LB/FIREWALL
VLAN2
106
INTERNET

Fig. 3

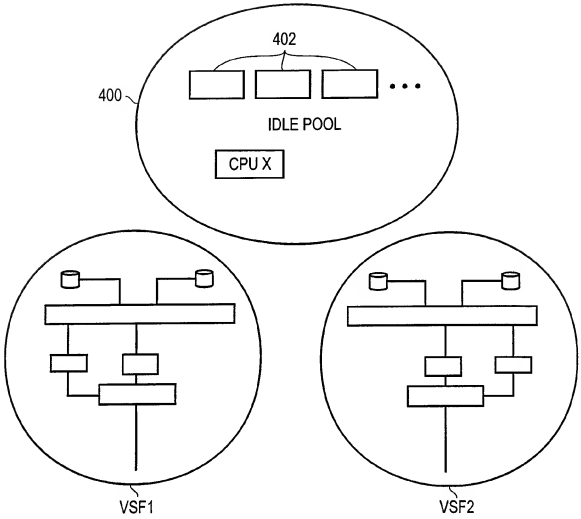


Fig. 4A

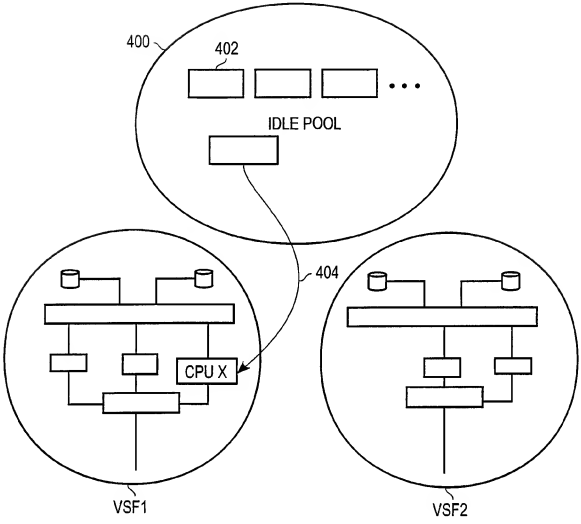


Fig. 4B

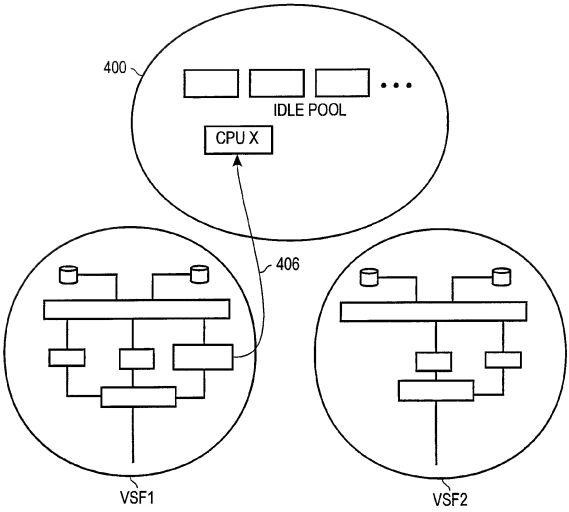


Fig. 4C

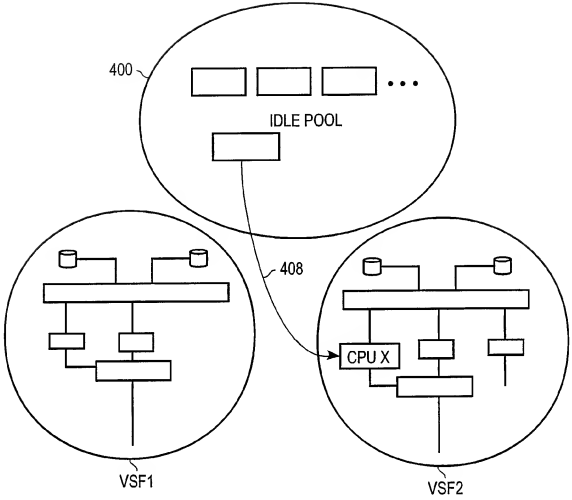


Fig. 4D

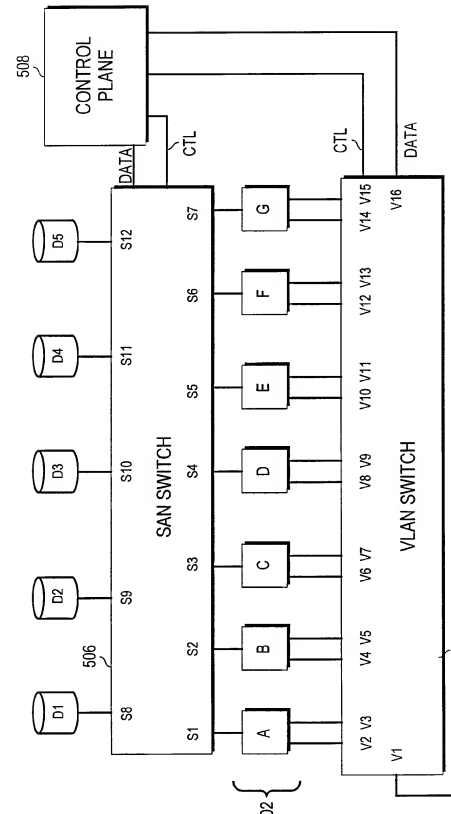

508
CONTROL PLANE
DATA
CTL
CTL
DATA
D1
D2
D3
D4
D5
506
SAN SWITCH
S1
S2
S3
S4
S5
S6
S7
S8
S9
S10
S11
S12
A
B
C
D
E
F
G
502
VLAN SWITCH
V1
V2 V3
V4 V5
V6 V7
V8 V9
V10 V11
V12 V13
V14 V15
V16
504
INTERNET
106

Fig. 5A

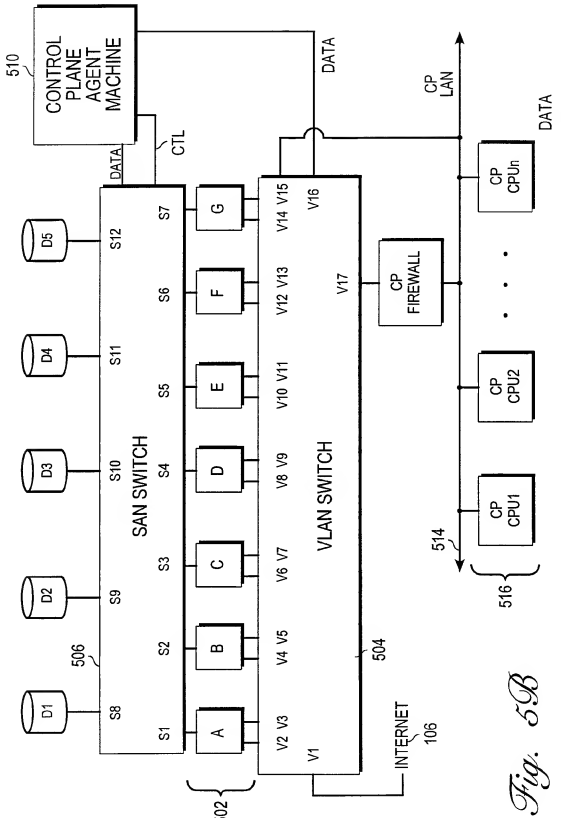

510
CONTROL PLANE AGENT MACHINE
DATA
CTL
DATA
CP LAN
DATA
D1
D2
D3
D4
D5
506
SAN SWITCH
S1
S2
S3
S4
S5
S6
S7
S8
S9
S10
S11
S12
A
B
C
D
E
F
G
502
VLAN SWITCH
V1
V2 V3
V4 V5
V6 V7
V8 V9
V10 V11
V12 V13
V14 V15
V16
V17
504
CP FIREWALL
INTERNET 106
514
516
CP CPU1
CP CPU2
CP CPUn

Fig. 5B

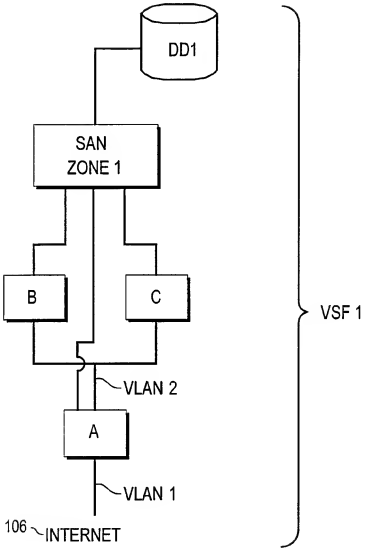


Fig. 6

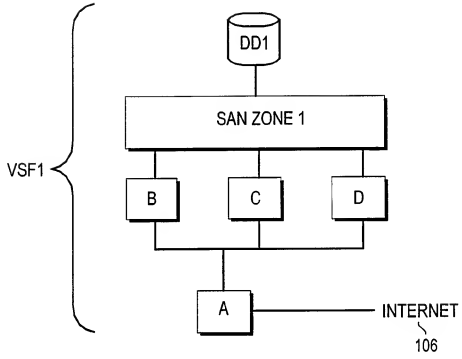


Fig. 7

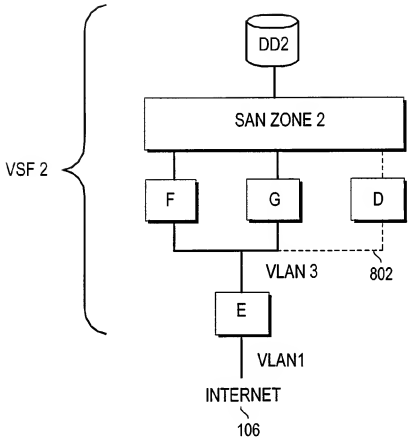


Fig. 8

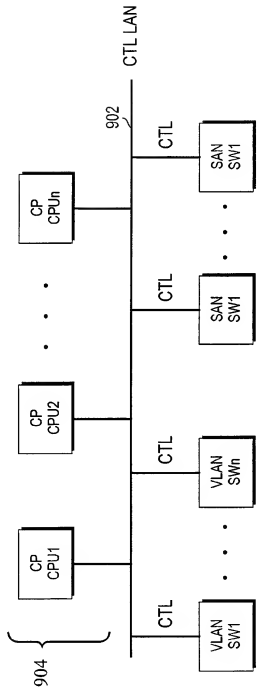

CTL LAN
902
904
CP CPU1
CP CPU2
CP CPUn
CTL
VLAN SW1
VLAN SWn
SAN SW1
SAN SW1

Fig. 9

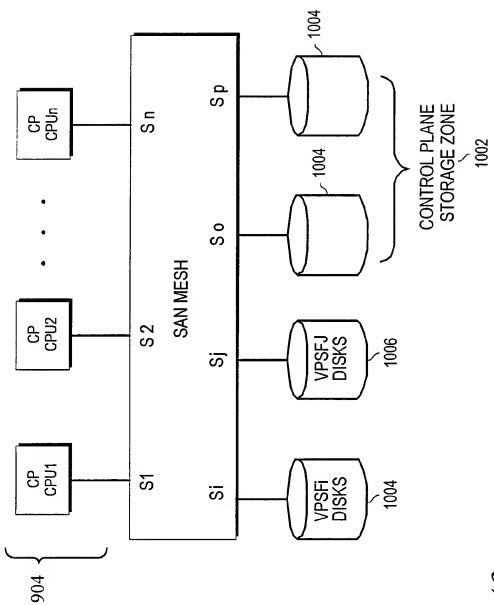

CP CPU1
CP CPU2
CP CPUn
904
S1
S2
Sn
Si
Sj
So
Sp
SAN MESH
VPSFi DISKS
1004
VPSFJ DISKS
1006
1004
1004
CONTROL PLANE STORAGE ZONE
1002

Fig. 10

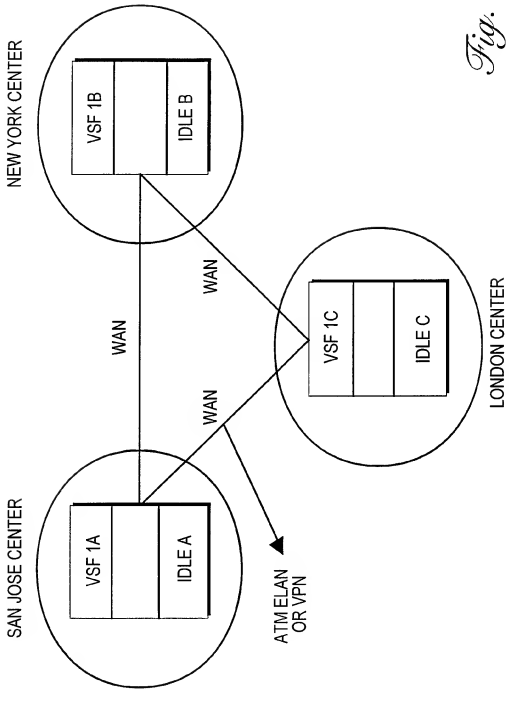

NEW YORK CENTER
VSF 1B
IDLE B
WAN
WAN
WAN
SAN JOSE CENTER
VSF 1A
IDLE A
ATM ELAN
OR VPN
VSF 1C
IDLE C
LONDON CENTER

Fig. 11

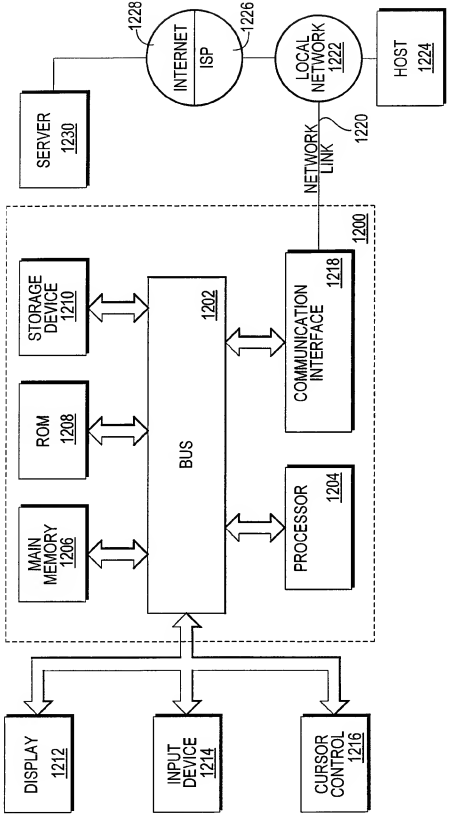

1228
INTERNET
ISP
1226
LOCAL NETWORK 1222
HOST 1224
SERVER 1230
NETWORK LINK
1220
1200
STORAGE DEVICE 1210
ROM 1208
MAIN MEMORY 1206
BUS 1202
COMMUNICATION INTERFACE 1218
PROCESSOR 1204
DISPLAY 1212
INPUT DEVICE 1214
CURSOR CONTROL 1216

Fig. 12